APPLICATION PROGRAMMING INTERFACE FOR ENABLING A DIGITAL TELEVISION RECEIVER TO ACCESS SYSTEM INFORMATION IN AN ABSTRACT FORMAT

BACKGROUND OF THE INVENTION

This application claims the benefit of U.S. Provisional Application Nos. 60/106,508, filed October 30, 1998, 60/107,965, filed November 12, 1998, and 60/113,444, filed December 23, 1998.

The following acronyms are used:

A/V - Audio/Video
API - Application Programming Interface
ATSC - Advanced Television Systems Committee
BAT - Bouquet Association Table (DVB)
CA - Conditional Access
CAT - Conditional Access Table (MPEG)
CNN - Cable News Network
DAVIC - Digital Audio-Video Council
DCII - GI Digicipher II (tm)
DIT - Data Information Table
DTV - Digital Television
DVB - Digital Video Broadcasting
DVS - Digital Video Standard
EIT - Event Information Table (DVB/ATSC)
EMM - Entitlement Management Message
EPG - Electronic Program Guide
ETT - Extended Text Table (ATSC)
FCC - Federal Communications Commission
GIC - General Instrument Corporation
GPS - Global Positioning Satellite
ID - Identifier

IP - Internet Protocol
JMF - Java Media Framework (Sun Microsystems)
MGT - Master Guide Table (ATSC)
MPAA - Motion Picture Association of America
MPEG - Moving Pictures Expert Group
MSP - Message Stream Protocol
NIT - Network Information Table (DVB)
NVOD - Near Video-On-Demand
PID - Packet Identifier
PMT - Program Map Table
PSI - Program Specific Information
PSIP - Program and System Information Protocol (ATSC)
RRT - Rating Region Table (ATSC)
SCTE - Society of Cable Television Engineers
SDT - Service Description Table (DVB)
SI - System Information
STT - System Time Table
TDT - Time Date Table (DVB)
TOT - Time Offset Table (DVB)
TS - Transport Stream
TSDT - Transport Stream Descriptor Table (MPEG)
UML - Unified Modeling Language
URL - Uniform Resource Locator
VCT - Virtual Channel Table
VSB - Vestigial Side Band

The present invention provides an API for accessing Program and System Information that describes the layout and content of an MPEG-2 TS. This

information, also known as service information, is generally called System Information (SI).

There are different formats of SI used and standardized today. These include the original ATSC A56 standard, which is a subset of the DigiCipher II (DCII) MSP, used for both satellite and cable television transmission, and the new ATSC PSIP for terrestrial and cable DTV, Cable SI such as DVS (SCTE DVS 234: Service Information Carried Out-Of-Band For Digital Cable Television), and the DVB SI standard. Private data, such as that in the DigiCipher II standard that is proprietary to GIC, the assignee of the present invention, may also be used.

The ATSC standard is described in "Program and System Information Protocol for Terrestrial Broadcast and Cable," Doc. A/65, 23 Dec 1997, available from the ATSC. The DVB standard is described in "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems," EN 300 468 v1.3.1 (1998-02), available from the European Broadcasting Union or the European Telecommunications Standards Institute.

Subscriber terminals receive the SI via a network. A set-top terminal, also referred to as an Integrated Receiver-Decoder (IRD) or a subscriber terminal, is a device that receives and decodes television signals for presentation by a television. The signals can be delivered over a satellite, through a cable plant, or by means of terrestrial broadcast, for example. Various applications have been proposed, or are currently available, via modern set tops, including

video on demand (VOD), audio on demand, pay-per-view, interactive shopping, electronic commerce, electronic program guides, Internet browsers, mail services (e.g., text e-mail, voice mail, audio mail, and/or video mail), telephony services, stock ticker, weather data, travel information, games, gambling, banking, shopping, voting, and others. Applications may also enable Internet connectivity and possibly Internet-based telephony. The set top functionality is enabled through specialized hardware and software.

The applications may be downloaded by terminals via a network, loaded locally (e.g., via a smart card), or installed at the time of manufacture, for example.

However, the subscriber terminal that receives the SI must known which format is being used, and provide corresponding processing that is specific to that protocol. This is problematic since it forces the development of special code (software) at the terminal for accessing the SI. Thus, the cost, complexity, and computational requirements of the terminals are increased, and the development of software for the terminal is impaired.

The above problems are highlighted by the trend toward integration broadband distribution networks, telephony networks, and computer networks such as the Internet and in-home networks, and by the desire to enable new types of applications that provide a feature-rich experience for the viewer.

Accordingly, it would be desirable to provide a system for accessing SI in a digital transport or other

data stream that is compatible with different SI formats.

The system should abstract common elements of the SI from the different formats to provide "abstract SI." The "abstract SI" should provide access to SI that is useful to an application at such a level of abstraction that the application does not have to be aware of what SI standard format is used to deliver the information to the receiver.

The system should avoid the need for the application to have special code (software) when it is intended to run in different environments, such as DVB, SCTE and ATSC-based systems.

The system should be suitable for use with SI that is provided using different formats, including MPEG Program Specific Information (PSI), Digital Video Broadcasting Service Information (DVB SI), Advanced Television Systems Committee Program and System Information Protocol (ATSC PSIP), Cable SI such as DVS, and private SI, such as those in the DCII system.

The system should allow different applications to retrieve only the specific SI they require.

The system should allow an application to retrieve a specific descriptor from the SI if needed.

A descriptor refers to a mechanism for extending table data. Generally, the various digital video standards allows the use of various types of tables of data for carrying SI. For example, a table may designate locations in a TS (e.g., PID, frequency) at which a particular channel or program is carried.

Moreover, since tables are fixed structures that are difficult to extend over time to accommodate additional information, descriptors have been developed as an extension mechanism. A descriptor can be appended in an outer loop or inner loop of the table. In an outer loop, the descriptor is appended at the end of the table and provides additional table entries each time the entire table is read. In an inner loop, the descriptor is appended at the end of a portion of the table and provides additional table entries each time that of the table is read.

A descriptor may comprise a tag followed by a field or string of information, for instance.

Descriptors are only included as needed, and do not interfere with receivers that have not been updated to recognize them.

Additionally, the system should make use of URL syntax concepts that are currently being defined for DTV.

The system should be implementable in an API at a subscriber terminal in a television network.

The API should be compatible with Java(tm), ActiveX(tm) or an equivalent type of component-based object-oriented technology.

The system should optionally provide asynchronous delivery of results, separation of MPEG-2 specific data from a TS, and incremental retrieval of SI data.

The system should be compatible with a URL locator syntax. Note that a URL definition for DTV is still being established. The current API definition supports

the DAVIC DVB URL and the GIC-proposed DTV and ATSC URLs.

The present invention provides a system having the above and other advantages.

SUMMARY OF THE INVENTION

The present invention provides an API that allows applications that are running on a digital television terminal to recover SI from a digital TS without regard to the specific format type. The API abstracts the relevant portions of the SI to provide it in a format that is usable by different applications at the terminal.

Additionally, a Descriptor sub-package described below allows an application to retrieve a specific DVB or ATSC or private (e.g., DCII) descriptor if it has a special need to do so.

This SI API definition further employs the URL concept that is well known in connection with Internet and browser applications. A formal URL definition for DTV is still under discussion. The current API definition supports the DAVIC DVB URL and the General Instrument Corporation (GI) proposed DTV and ATSC URLs.

A television set-top terminal in accordance with the invention includes a computer readable medium (e.g., such as a magnetic or optical storage device) having computer program code means (e.g., object-oriented code such as Java(tm)), and means for executing (e.g., any processor such as a CPU) the computer program code means to implement an Application Programming Interface (API).

The API is adapted to abstract SI in a digital television transport stream that is received by the terminal in any one of a plurality of different formats. The API provides the abstracted SI in a

generic format that is suitable for use by an application at the terminal regardless of the specific format in which the SI is provided. For example, the different SI formats may include different data table formats.

This allows a terminal to be compatible with a number of different data stream formats.

Moreover, the API may provide a number of functions at the terminal that are responsive to the abstracted SI, such as a navigation function that allows the terminal to navigate among television channels in the transport stream, a program guide function that implements an electronic program guide for the television channels, a selection function that selects specific television channels, and a descriptor retrieval function for recovering descriptors of the SI.

Additionally, the API provides a utility function containing supporting objects, including events and exceptions, for supporting the delivery of the SI to the application synchronously, a data function for implementing a guide to data services in the transport stream in accordance with the abstracted SI, and a pipeline function for providing information regarding a physical delivery mechanism (e.g., satellite or transponder identifier) of the transport stream. The pipeline function uses the appropriate SI tables to provide information about the delivery network (MPEG-2 Transport Stream, etc.).

The API provides these functions by exposing the appropriate SI data to the application. An application

such as an EPG can use these APIs to do its job, e.g. an EPG uses the Navigation package (or function) to learn about channels, and it uses the Guide package (function) to learn about scheduled programs on the selected channels. Thus, the APIs or packages discussed herein are used (called) by specific applications.

The different available SI formats can include: Motion Picture Experts Group (MPEG) Program Specific Information (PSI), Digital Video Broadcasting (DVB) System Information (SI), Advanced Television Systems Committee (ATSC) Program and System Information Protocol (PSIP), Cable SI Digital Video Standard 234 of the Society of Cable and Television Engineers, and private SI.

The terminal may include a memory for storing the SI as the transport stream is received at the terminal, where the API provides a retrieve function call for enabling a calling application to retrieve the SI such that SI that is available in the memory is returned essentially immediately as a direct return value. If the service information is not available in the memory, the retrieve function call returns an exception signaling to the calling application that the SI is to be delivered to the calling application asynchronously. In this case, the API may also provides a utility function containing supporting objects, including events and exceptions, for supporting the asynchronous delivery of the SI to the calling application.

Moreover, when the transport stream is provided in one of a plurality of available transport stream

formats, the API may abstract the SI to provide it in a generic format that is suitable for use by an application at the terminal. This may be achieved by providing a base package having information that is generic to the available transport stream formats. The API is adapted for use with a separate package having information that is specific to the format of the received transport stream.

Generally, the API provides the base set of APIs, which is extensible. For instance, ATSC is adding new packages to provide PSIP specific info, and DVB may do the same. Such packages are not shown here. However, the API of the present invention is extensible so that format-specific extensions can be easily made by subclassing or extending the Abstract SI API classes and interfaces.

In a further aspect of the invention, the API provides incremental retrieval of the service information by allowing a calling application at the terminal to obtain a subset of the SI that is available at the terminal, perform an analysis of the obtained SI, and retrieve additional SI if required based on the analysis. The additional SI may be retrieved from the subset of the SI that is available at the terminal in a memory of the terminal, or from the transport stream.

Note that the SI data may be stored at the terminal using a variety of implementations. For example, it can be stored before and/or after abstracting, after optimizing and compression, and so forth.

The API enables a calling application at the terminal to recover a subset of the SI in the transport stream while rejecting other SI in the transport stream that is not required by the calling application.

The API may also provide a filtering function that is responsive to the abstracted SI to allow the application to specify at least one service in the transport stream in which the application is interested. The filtering can be based on whether the services are associated with: a particular transport stream (when services from multiple transport streams are available - a receiver may have multiple tuners, or SI data may be stored in the receiver's memory, in a SI database, that is collected from multiple transport streams over time), a network, a bouquet, a satellite, a satellite transponder, a service name, a service/channel number, a favorite channel, and a theme.

Moreover, the API may be implemented using a plurality of packages for abstracting the SI, in which case it is efficient for the different applications at the terminal to include only specific ones of the packages according to specific portions of the abstracted SI that each application requires.

A corresponding method is also presented.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows package relationships and dependencies of an API in accordance with the present invention.

FIG. 2 illustrates a navigation package class/interface diagram in accordance with the present invention.

FIG. 3 illustrates a program guide package class/interface diagram in accordance with the present invention.

FIG. 4 illustrates a selection package class/interface diagram in accordance with the present invention.

FIG. 5 illustrates a descriptor package class/interface diagram in accordance with the present invention.

FIG. 6 illustrates a pipeline package class/interface diagram in accordance with the present invention.

FIG. 7 illustrates a data package class/interface diagram in accordance with the present invention.

FIG. 8 illustrates a utility package class/interface diagram in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

An API allows applications that are running on a digital television terminal to recover SI from a digital TS without regard to the specific format type. The API abstracts the relevant portions of the SI to provide it in a format that is usable by different applications at the terminal.

The API is preferably independent of an operating system and hardware of the terminal.

Note that the figures were generated automatically from Rational Rose(tm) CASE tool, developed by Rational Software Corporation, USA. The figures use the Rational Rose (tm) depiction of the UML, which is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. A class diagram represents the static structure of a system, and shows a pattern of behaviors that the system exhibits. This is accomplished by showing the existence of classes and their relationships. Each class is represented by a box with three sections. The top section lists the class name. The middle section denotes a list of attributes, and the bottom section denotes a list of operations.

A solid or dashed line between classes denotes an association or dependency. A white diamond tip denotes aggregation by reference, while a black diamond tip denotes aggregation by value. A triangular arrowhead denotes a restricted navigation, e.g., inheritance of operation but not of structure.

Moreover, interfaces and classes begin with an uppercase letter, while methods begin with a lowercase letter.

A class is a template that defines a data structure, method and function calls for an object. An interface defines a set of methods/function calls that can be manipulated by a class. The class provides the code for implementing an interface.

1. Model Description

The entire SI database model is based on the concept of "views". There are different ways to look at the SI database, and different applications may have different needs. The specified views allow applications to be concerned only with a subset of the SI database based on the application's needs. The concept of views is represented by Java packages.

The following sections describe the object model of the Abstract SI API.

FIG. 1 is a high-level figure that shows package relationships and dependencies of an API in accordance with the present invention.

The Navigation package 110 contains the main set of classes and interfaces that are used to navigate the existing television channels (DVB Services or ATSC Virtual Channels). The Selection package 130 adds features which allow a TvChannel to be used for tuning and selection of specific services (MPEG programs). The Guide package 120 provides information useful for an EPG, including program schedules, individual program events and program ratings. The Descriptor package 140

allows a retrieval of individual MPEG descriptors associated with the MPEG sections that deliver the SI. The Pipeline package 170 isolates some of the specific delivery media information. The classes provided in this package currently represent the MPEG-2 delivery mechanism. The Data package 160 is similar to the Guide package 120 except it provides information about data-related events (services), e.g., any software application that is sent along with the audio/video stream, such as a stock ticker, news ticker, sports statistics, interactive commercial, etc.), not audio/video events. Lastly, the Util (Utility) package 150 contains objects of a supporting nature, such as events, exceptions, etc.

This model provides both a high-level abstraction of the SI that describe the layout of the content delivered over the multiplex (meta-data), as well as detailed information specific to a particular SI format. This is done via access to MPEG table descriptors. Since descriptors are used as way to extend the SI for additional and future functionality, it is a significant advantage of the invention that the API provides a generic access to the descriptors without changing the implementation of the API with every new or changed descriptor.

In one aspect of the invention, asynchronous delivery of results is provided. Since there are DTV receivers with varying capabilities, it is expected that not every receiver will cache all the SI data in the memory. It will do so for the necessary subset of the most useful information, but will have to parse the

actual TS when it needs to retrieve data not stored in memory. However, access to the TS may take a significant amount of time. Therefore the API provides an asynchronous access to information that is not expected to be in memory at all times. In general, asynchronous (or non-blocking) calls add complexity not only to the implementation of the API but to the application using the API as well.

To hide the difference between low-end and high-end receivers, this API provides a single method which can be completed synchronously, if the information is available locally (in memory), or asynchronously, if the data have to be retrieved from the TS. All API calls starting with "retrieve" either return the requested object or throw an exception indicating that the data will be delivered later via an asynchronous event. The caller can register itself as an listener to this event. Or, by not registering, the caller can indicate that it is interested in synchronous data delivery only. The exception includes enough information to cancel the request and to associate it with the actual event delivering the data.

The SI usually contains system time related information in the form of the PSIP STT message or DVB TDT and TOT messages. At this time it is assumed that the local DTV receiver time will be synchronized with the system time of the channel currently tuned to and the value can be obtained using the java.util.Date class. Note that this is acceptable if all transport multiplexes use a reliable and synchronized data source (such as GPS). If a certain multiplex provides

erroneous time information, it is up to the receiver implementation to resolve the situation.

1.1 Navigation View

FIG. 2 illustrates a navigation package class/interface diagram in accordance with the present invention.

Like-numbered elements correspond to one another in the figures.

The Navigation package 110 has two main functions, namely:

(1) give access to all or a selected subset of TvChannels, which represent DVB services and ATSC virtual channels; and

(2) give access to network related information such as the network definition, satellite and transponder information, TS and bouquet information, etc.

Like-numbered elements correspond to one another in the figures.

The package includes the following classes and interfaces: CAIdentification 205, DeliverySystemType 807, SystemInformationType 810, ChannelCollection 220, TvChannel 225, SIUpdate 830, ChannelConstraint 235, SIManager 240, TvChannelDetails 245, and ChannelIdentification 250.

The main navigation function is represented by the following classes and interfaces. The SIManager 240 is the primary access point to the underlying SI database. It can generate a collection of TvChannels 225 called ChannelCollection 220 based on the selection criteria

represented by the ChannelConstraint 235 object. The selection criterion may be a network ID, TS ID, bouquet ID, user favorite channels, URL, etc. The collection can then be used to sort either by channel numbers or by channel names and navigate through the TvChannels 225 which represent either a DVB Service or an ATSC Virtual Channel.

The TvChannel 225 itself contains only the minimal information (such as Locator, Channel name and number) needed to navigate. Additional information about the channel is contained in the TvChannelDetails object 245. The TvChannelDetails also provides some CA-related information via the CAIdentification interface 205, a delivery mechanism, and the time when the information about this channel was last updated. The channel name and number is actually encapsulated in the ChannelIdentification object 250, which can be extended to accommodate different mechanisms for naming and numbering channels, such as the ATSC two-part channel number (e.g., using a major.minor notation - a string with a dot between the major and minor channel numbers - ex: 10.2).

1.2 Guide View

FIG. 3 illustrates a program guide package class/interface diagram in accordance with the present invention.

The package includes the following interfaces: SIManager 240, TvChannelDetails 245, TvChannelWithSchedule 300, ContentRatingAdvisory 310, ProgramSchedule 320, RatingRegion 330, ProgramEvent

340, RatingDimension 350 and SIUpdate 830 (defined in FIG. 8).

This package is intended to support electronic program guide type applications. It provides the application with two related sets of information: the program schedule on each channel, and rating information. The ProgramSchedule object 320 can be used to retrieve the currently-playing program, the immediately following one, and then any other available program in the future for a specified time period. Each ProgramEvent 340 can be queried for its name, start time and end time, description, rating, cost and other related information.

Rating related information is organized into rating regions, where each region may have a multiple rating dimensions such as MPAA rating, FCC TV rating, DVB age-based rating, etc. Each dimension contains multiple levels; each ProgramEvent is labeled with one of these levels for all supported rating regions.

1.3 Selection View

FIG. 4 illustrates a selection/tuning package class/interface diagram in accordance with the present invention.

The package includes the following classes and interfaces: TvChannelDetails 245, ChannelComponent 410, MPEGChannelComponent 420, Locator 430, MPEGLocator 440 and ATSCLocator 450.

The tuning package extends the navigation functionality primarily by adding information about individual components of each TvChannel, such as video,

possibly multiple audio components and optional data components. The TvChannel object 225 can provide a Locator used by a TV Player which follows the JMF Player model or a separate service selection API.

The generic ChannelComponent 410 is extended in by the MPEGChannelComponent 420, which represents MPEG-specific information.

The basic Locator 430 concept is extended to support the MPEG-specific Locator (MPEGLocator 440), which is then specialized for ATSC (ATSCLocator 450). Note that the DAVIC package includes a DVB-specific subclass as well.

1.4 Descriptor View

FIG. 5 illustrates a descriptor package class/interface diagram in accordance with the present invention.

The package includes the following interfaces: TableType 510, DescriptorTag 520, Descriptor 530, MPEGTableDescriptors 540 and MPEGPipeline 660 (defined in FIG. 6).

Descriptors are generally delivered in MPEG-2 tables in two locations: the outer loop which associates descriptors with the entire table, and the inner loop which associates descriptors with the specific entity described in the inner loop. An example is a PSIP VCT, which has both outer- and inner-loop descriptors. The outer loop descriptors are associated with all virtual channels defined by this VCT, while each inner loop contains descriptors for a specific virtual channel.

Applications executing on a set-top will use APIs to access the set-top functions including the SI. The goal is to provide a format-independent API to retrieve SI (ATSC PSIP, DVB SI, etc.) to minimize the application's required knowledge of the details of these SI formats. Abstraction is good in most cases, but certain applications need access to a specific descriptor which is not provided at the abstract API level. For example, this can be a descriptor which will be defined in the future. Therefore, it cannot be represented in the API directly. It can also be one of the descriptors that are not intended for an application; they are rather used by the receiver itself (e.g., AC3 descriptor, Linkage descriptor, etc.)

The problem is to specify the specific descriptors an application is interested in. This applies to both determining the appropriate table, as well as the position of the descriptor loop.

The type of information (e.g., TS, service, event, etc.) for which descriptors are to be retrieved is primarily identified by a DTV URL. There are some rare exceptions; e.g., the DVB BAT is really not identified by any current URL formats. The DTV URL is usually sufficient to point to a specific <table_type, descriptor loop> pair, especially for URLs pointing to an event or an elementary stream.

There are several cases where the URL is not specific enough because an entity, a service for instance, may be described in multiple tables, such as PMT and DVB SDT. Therefore, a table_type is specified to identify which type of a table to retrieve

descriptors from. Table_id is not used since some table types have different table_ids within the type (e.g., DVB NIT has 0x40 and 0x41). The table_type scopes down the range of tables to search for descriptor especially at the higher level of URLs (the service and TS levels).

In some rare cases, an additional identification is needed. For instance, if a rating_type is specified, a Rating Region ID is needed to retrieve the proper subset of descriptors from the ATSC RRT. The same applies to navigation_type tables where a Bouquet ID is needed to retrieve descriptors from the DVB BAT outer descriptor loop.

Additionally an optional set of descriptor tags can be specified in the call to limit the search and the returned set of descriptors to the descriptors hinted in the list. If a set of descriptors is found in the given table identified by the <url, table_type, entity_info> tuple (e.g., set of values), only those matching the descriptor tags provided in the hint will be returned.

Note that all descriptor retrieval methods support both synchronous as well as asynchronous data delivery depending on the caching capability of the DTV receiver.

1.5 Pipeline View

In an optional embodiment, separation of MPEG-2 specific data is provided.

Digital television content is primarily delivered using the MPEG-2 transport format. This is true for

DVB as well as ATSC. With the convergence of television and personal computers, it is expected that digital video content may be delivered to the television receiver by other means, such as over the Internet in an IP format with some kind of a real-time protocol. This motivates a higher level of abstraction for the SI API that hides not only the difference between ATSC PSIP and DVB SI (both extensions of MPEG-2), but also the difference between the different ways of delivering the content and the SI.

This design removes all MPEG-2 (e.g., TS) specific information into a separate package. A base class, which provides generic transport-neutral information, can be further extended with MPEG-2 specific information for MPEG-2 delivery networks. As new delivery mechanisms become popular, the base classes can be extended to provide detailed information about the specific protocol.

Essentially, the API further abstracts the SI to provide it in a manner that is independent of the specific TS formats. The API can therefore run on a terminal that receives a TS in any one of a number of different available formats.

FIG. 6 illustrates a pipeline package class/interface diagram in accordance with the present invention.

The package includes the following classes and interfaces: SIManager 240, SatelliteInformation 605, NetworkInformation 610, BouquetInformation 630, SISpecificManager 640, PipelineInformation 650,

TransportStreamInfo 660, MPEGPipeline 670 and SystemInformationType 810 (defined in FIG. 8).

The Pipeline package provides additional information about the physical mechanism media delivering the content the SI data describes. The physical delivery mechanism can include, e.g., satellite/transponder information. SISpecificManager 640 provides access to the abstract PipelineInformation 650, which in this particular case is extended by the MPEGPipeline 670 representing an MPEG-2 multiplex. The generic PipelineInformation 650 can be extended to support other types of delivery of content (e.g., Internet Protocols).

1.6 Data View

FIG. 7 illustrates a data package class/interface diagram in accordance with the present invention.

The package includes the following interfaces: TvChannelDetails 245, ContentRatingAdvisory 310, TvChannelWithData 710, DataSchedule 720, DataEvent 730 and SIUpdate 830 (defined in FIG. 8)

The Data view is similar to the Guide view, which represents EPG-like information. In this case, the data schedule represents a lineup or guide of data events as opposed to audio/video events.

Note that this package is modeled after the ATSC T3/S13 work, which is still in progress.

1.7 Utility View

FIG. 8 illustrates a utility package class/interface diagram in accordance with the present invention.

The package includes the following classes and interfaces: SIFactory 805, SIManager 240, SystemInformationType 810, SIChangeListener 815, EventObject 820, SIUpdate 830, SIChangeEventType 832, SIChangeEvent 834, TvChannelChangeEvent 835, DataChangeEvent 840, ProgramChangeEvent 845, SIInfoChangeEvent 850, Exception 860 (from the java.lang package), SIRetrievalEvent 865, SIRetrievalListener 870, SIDelayedDeliveryException 875, SIException 880, SIRetrievalFailEvent 882, SIRetrievalSuccessEvent 884, SIRequest 886, SINotAvailableException 890, SIRetrievalSingleSuccessEvent 892 and SIRetrievalMultipleSuccessEvent 894.

The utility package provides support in several areas, namely:

1. the event notification mechanism for both SI entity changes detected in the TS and events delivering asynchronous requests;

2. the SI Factory which creates SI Manager(s); and

3. exceptions.

The SIChangeListener 815 and the SIChangeEvent 834 support the standard Java event model. The are three types of objects to listen to for changes:

1. The SIManager 240, which reports changes detected in the Network definition related tables represented by the TransportStreamInfo 660,

NetworkInformation 610, BouquetInformation 630, etc. objects.

2. The ChannelCollection 220, which reports changes detected in any one of the TvChannels 225 contained by the collection.

3. The ProgramSchedule 320, which reports changes detected in any one of the ProgramEvents 340 in the schedule.

Applications can register as listeners with the above-listed objects, and they will be notified by receiving one of the three appropriate events, which will deliver the details about which specific object has changed. To obtain the new information, the application needs to regenerate the particular collection of objects (ChannelCollection 220, ProgramSchedule 320, etc.).

Note that it may be difficult to implement the above defined events unless the receiver does a field-by-field comparison of old and new tables of the same type. The receiver may choose to deliver only the high-level event and let the application update multiple objects if necessary.

The Utility package 150 also provides the mechanism to deliver data asynchronously. This functionality is provided by the SIRequest 886, SIDelayedDeliveryException 875, SIRetrievalEvent 865 and SIRetrievalListener 870. The SIDelayedDeliveryException 875 signals to the caller that the data is available only asynchronously and provides the SIRequest object 886 which can be used to cancel the request and to associate this request with

the SIRetrievalEvent 865, which eventually delivers the requested data or an indication of a failure. The caller of the asynchronous method (indicated by the "retrieveXXX" name) may register as a listener to get the event. If it decides not to register as a listener, it is an indication to the API implementation that there is no need to parse the TS and retrieve the requested data. Thus, in accordance with the present invention, the application has complete control over the delivery of data.

Thus, incremental retrieval of SI data can be provided. Generally, there is a wide variety of applications that will execute on DTV receivers. Some of the receivers need access to the full SI data set (such as EPG-like applications), while others may need only a very small subset of the SI data. To support all of these applications without putting an extra burden on those needing a small subset of the SI data, the present invention enables incremental retrieval of SI data. This allows an application to obtain a small set of SI data, make an intelligent decision, and retrieve more SI for possibly a selected SI object or a subset of SI objects.

Such a design provides flexibility, more control to the application, and more efficient retrieval of SI data.

Since most of the SI objects are really interfaces which don't have constructors, an application does not have a way to instantiate an object that implements the specified interface. To get an instance of the object that implements the SIManager interface 240, an

SIFactory class 805 is provided which has a method to obtain an instance of the SIManager.

Several methods throw Exceptions to report invalid parameters or other types of error conditions (see classes 875 and 880).

2.0 Class and Interface Description

2.1 Navigation Package (FIG. 2)

This view at the SI is from the point of view of a navigation mechanism, such as a simple channel guide, channelUp and channelDown buttons, etc.

It provides enough information to present a list of available MPEG2 services with a filtering mechanism.

2.1.1 TvChannel 225

TvChannel represents an abstract view on what is referred to as an MPEG Program, DVB Service or an ATSC Virtual Channel. It represents the common information associated with it, such as channel name, channel number, description, etc. Each TvChannel is uniquely identified by a tuple including system type, network id, TS ID, service number or channel number. This identification may be represented in the URL format.

Public Operations:

getLocator () : Locator

Returns the complete Locator of this TvChannel.

getLongChannelName () : String

Returns a full channel name.

isHidden () : boolean

Returns TRUE is this is a hidden channel.

retrieveDetails (listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.navigation.TvChannelDetails

This method retrieves additional information about the TvChannel. This information is based on the SI data (also called meta-data). SI data can be considered to be information about other data (content) such as audio/video/data components of the transport stream. Since it is data about other data, it is sometimes called meta-data.

This method may return data synchronously or asynchronously.

getIdentification () : org.atsc.abstractSI.navigation.ChannelIdentification

This method is used to obtain the channel identification (e.g., channel name and number).

2.1.2 ChannelCollection 220

ChannelCollection represents a collection of TVChannels 225 based on a specific grouping rule defined by the ChannelConstraint association class 235. Filtering used to create such a collection may be based on TS ID, Network ID, System Type (DVB, ATSC, etc.), Bouquet, content theme (e.g., sports), Channel name or a subset of it (e.g., CNN) or possibly a combination of these.

This class also provides a mechanism for browsing those TVChannels contained by the particular collection instance.

This is similar to the SortedMap interface (from Sun Microsystem's JDK 1.2 APIs) but provides only a

small subset of the functionality as applicable to this domain.

Public Attributes:

SORT_BY_CHANNEL_NUMBER : short = 1

SORT_BY_NAME : short = 2

Public Operations:

size () : int

Returns the number of TvChannels contained in this collection.

sort (criterion : short) : void

Called to specify an algorithm for determining the behavior of the nextChannel and previousChannel methods.

For example, if sorting by channel name is specified, the nextChannel method will return the next TvChannel object with a name alphabetically following the current TvChannel. It is always sorted in an ascending order.

nextChannel (currentChannel : org.atsc.abstractSI.navigation.TvChannel) : TvChannel

Returns the next TvChannel relative to the specified TvChannel based on the sorting criterion. Null is returned if end of collection is reached.

previousChannel (currentChannel : org.atsc.abstractSI.navigation.TvChannel) : TvChannel

Returns the previous TvChannel based on the sorting criterion. Null is returned if the beginning of this collection is reached.

firstChannel () : TvChannel

Returns the first TvChannel in this collection based on the sorting criterion set for this collection.

lastChannel () : TvChannel

Returns the last TvChannel in this collection based on the sorting criterion set for this collection.

addListener (listener : SIChangeListener) : void

Called to register an SIChangeListener 815 for changes related to the Channels in this collection. A TvChannelChangeEvent 835 will be delivered to the listener.

removeListener (listener : SIChangeListener) : void

Called to deregister a SIChangeListener for changes related to the Channels in this collection.

retrieveChannels (filter : org.atsc.abstractSI.navigation.ChannelConstraint, listener : org.atsc.abstractSI.util.SIRetrievalListener) : ChannelCollection

This method returns a ChannelCollection object 220, which is a subset of this collection, based on the grouping conditions specified in the filter parameter. If the filter is null, a collection of all TvChannels 225 contained in this collection is returned.

This method is provided to generate an increasingly specialized collections of TvChannels based on multiple filtering (grouping) criteria.

findChannel (locator : org.davic.net.Locator) : org.atsc.abstractSI.navigation.TvChannel

This method returns the TvChannel corresponding to the specified locator if it is a member of this collection. Otherwise, it returns null.

findChannel (channelID : org.atsc.abstractSI.navigation.ChannelIdentification) : org.atsc.abstractSI.navigation.TvChannel

This method returns the TvChannel corresponding to the specified channel identification if it is a member of this collection. Otherwise, it returns null.

2.1.3 SIManager 240

SI Manager represents the central managing entity which has a knowledge of the entire network or a collection of networks, and can create a collection of TvChannels based on the ChannelConstraint filtering rules.

It has also access to specific SI representations of each individual TS (DVB SIDatabase, ATSC PSIPDatabase, etc.)

Public Operations:

retrieveChannels (filter : org.atsc.abstractSI.navigation.ChannelConstraint, listener : org.atsc.abstractSI.util.SIRetrievalListener) : ChannelCollection

This method returns a ChannelCollection object based on the grouping conditions specified in the filter parameter. If the filter is null, a collection of all known TvChannels is generated.

This method can deliver results both synchronously or asynchronously. If requested data is available immediately, it is returned synchronously.

If the data must be retrieved from the transport first, the SIDelayedDeliveryException 875 is thrown and

the results are delivered to the registered listener via an event.

Parameter filter - A rule constraining the requested channel collection.

Parameter listener - A listener which receives the delivery event when data are retrieved asynchronously. If an application does not provide a listener (null), no asynchronous retrieval is attempted. The listener is registered for this one call only.

setPreferredLanguage (language : int) : void

This method sets the language used to return any textual information from the SI related classes and interfaces (e.g., TvChannel name, etc.) if provided as a multilingual string in multiple languages. If the specified language is not available, the system-level preferred language is used. If that language is not available either, the first available language will be used.

This method is used to temporarily override the system-level preferred language within the abstractSI package.

getPreferredLanguage () : int

This method is called to determine the preferred language for returning string-type values.

getRatingRegions () : int[]

This method returns a list of available Rating Region IDs.

retrieveRatingRegion (regionID : int, listener : org.atsc.abstractSI.util.SIRetrievalListener) : RatingRegion

This method is used to obtain a RatingRegion object of the specified ratingRegionID. This method may deliver data synchronously as well as asynchronously.

2.1.4 ChannelConstraint 235

This association class represents a set of rules or filtering criteria used to generate a particular ChannelCollection 220. Filtering used to create such a collection may be based on TS ID, Network ID, System Type (DVB, ATSC, etc.), Bouquet, content theme (e.g., sports), Channel name or a subset of it (e.g., CNN), etc.

Public Operations:

ChannelConstraint (filter : int, value : java.lang.Object) :

This constructor specifies what the grouping criteria should be.

Parameter filter - Filter represents an enumerated value of a specific filter type.

Parameter value - Sets the filter value based on the filter type.

getFilterType () : int

Called to determine what grouping mechanism is used for this ChannelConstraint.

getFilterValue () : java.lang.Object

Called to determine the value of the current filter. The meaning of the value changes based on the filter type.

2.1.5 CAIdentification 205

The CAIdentification interface provides a mechanism to associate CA-related information with any SI-related class, such as a TS or a TvChannel. It accesses information found in the CAT MPEG table.

Public Operations:

getCASystemIDs () : int[]

Returns an array of CA System IDs as defined in the CAT MPEG message. Returns null if no CAT information is provided for this channel.

isAccessControlled () : boolean

Returns TRUE if this TvChannel is protected by CA. Returns FALSE if it is not protected or unknown.

2.1.6 FilterType 265

This interface provides a definition of constant values of supported filtering mechanisms, such as filtering by Network ID, TS ID, etc.

Public Attributes:

NETWORK_ID_FILTER : short = 1

Filter based on Network ID

TRANSPORT_ID_FILTER : short = 2

Filter based on TS ID

BOUQUET_ID_FILTER : short = 3

Filter based on Bouquet ID

SYSTEM_TYPE_FILTER : short = 4

Filter based on SI type (e.g., DVB, ATSC, etc.)

SATELLITE_FILTER : short = 5

Filter based on Satellite ID

TRANSPONDER_FILTER : short = 6

Filter based on Transponder Number

LOCATOR_FILTER : short = 7

Filter based on a Locator (URL)

CHANNEL_NAME_FILTER : short = 8

Filter based on channel names

CHANNEL_NUMBER_FILTER : short = 9

Filter based on channel numbers

THEME_FILTER : short = 10

Filter based on themes/content categories

FAVORITE_CHANNELS_FILTER : short = 11

Filter based on user favorite channels

2.1.7 TvChannelDetails 245

This interface provides access to TvChannel metadata.

Derived from DeliverySystemType 807, SIUpdate 830, SystemInformationType 810, and CAIdentification 205.

Public Operations:

retrieveChannelDescription (listener : org.atsc.abstractSI.util.SIRetrievalListener) : String

Returns a textual description of this channel or null if none available.

getServiceProvider () : String

Returns the name of the service provider.

getServiceType () : int

Returns the type of this service. Service type is one of the following values: digital TV, digital radio, analog TV, analog radio, data service, NVOD reference service, NVOD time shifted service. This list may be extended with new types of services in the future.

retrieveComponents (listener : org.atsc.abstractSI.util.SIRetrievalListener) : ChannelComponent[]

This method returns a list of elementary components which are part of this channel.

getRunningStatus () : short

Returns the running status of this service (see DVB SI documentation).

retrieveContentAdvisory (listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.guide.ContentRatingAdvisory[]

Returns a list of Content Advisory information for each Rating Region.

getDeliverySystemType () : int

Called to determine the mechanism of delivering this TvChannel (e.g., cable, satellite, etc).

2.1.8 ChannelIdentification 250

This interface is used to provide a flexible and extensible way of identifying TvChannels by names, channel numbers, or other means.

It can be extended to support specific mechanisms such as the ATSC two-part channel numbers.

Public Operations:

getChannelNumber () : String

Returns a channel number, which is a system-specific value. For example, DCII uses a single number, DVB does not really support channel numbers or they are set-top or broadcaster specific, and ATSC now has the two-part channel number.

In the ATSC domain, it is in the major.minor notation (a string with a dot between the major and minor channel numbers).

For DVB channels, it can be the service ID as the channel number.

It can also represent a broadcaster-specific channel numbering mechanism delivered as a private descriptor.

getChannelName () : String

Returns a short channel name or an acronym.

2.2 Guide Package (FIG. 3)

This view of the SI is from the point of view of a program guide which shows not only the service availability but a list of future events offered on each of them.

2.2.1 ProgramSchedule 320

This interface represents a collection of program events for a given TvChannel 225 ordered by time. It provides the current, next and future events.

Public Operations:

retrievePresentEvent (listener : org.atsc.abstractSI.util.SIRetrievalListener) : ProgramEvent

Returns the current (can be viewed if tuned to) program event.

retrieveFollowingEvent (listener : org.atsc.abstractSI.util.SIRetrievalListener) : ProgramEvent

Returns the program event which immediately follows the current program event.

retrieveFutureEvent (when : java.util.Date, listener :

org.atsc.abstractSI.util.SIRetrievalListener) : ProgramEvent

Returns the program event for the specified time. The program event which contains the specified time will be returned. The specified time falls between the program event's start time and the start time plus the event's duration.

Null is returned when the specified time does not fall inside any known program event.

retrieveFutureEvents (startOfInterval : java.util.Date, endOfInterval : java.util.Date, listener : org.atsc.abstractSI.util.SIRetrievalListener) : ProgramEvent[]

Returns all known program events on this channel for the specified time interval.

retrieveEvent (locator : org.davic.net.Locator, listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.guide.ProgramEvent

This method retrieves a program event matching the locator. Note that the event must be part of this schedule.

addListener (listener : org.atsc.abstractSI.util.SIChangeListener) : void

Called to register an SIChangeListener 815 for events related to changes of the ProgramEvents 340 on this schedule. The ProgramChangeEvent 845 is delivered to the specified listener when any ProgramEvent on this schedule changes.

removeListener (listener : SIChangeListener) : void

Called to deregister a SIChangeListener 815.

2.2.2 ProgramEvent 340

This interface comprises a collection of elementary streams with a common time base, an associated start time, and an associated end time. An event is equivalent to the common industry usage of "TV program."

The Event Information Table (EIT) contains information (titles, start times, etc.) for events on defined TvChannels. An event is, in most cases, a typical TV program, however its definition may be extended to include particular data broadcasting sessions and other information segments, such as an infomercial, or to show that part of the event includes an interactive data application and the other part does not.

Derived from SystemInformationType 810 and SIUpdate 830. SIUpdate is an interface which is shared by all SI objects which represent SI tables and it provides information about the last time this object was updated.

Public Operations:

getLocator () : Locator

Returns a Locator representing this program event.

getStartTime () : java.util.Date

Returns the start time of this program event.

getEndTime () : java.util.Date

Returns the end time of this program event.

getDuration () : long

Returns the duration of this program event in seconds.

getEventName () : String

Returns the program event title.

retrieveDescription (listener : org.atsc.abstractSI.util.SIRetrievalListener) : String

Returns a textual description of the event. This information comes from the Extended Text Table (ETT) in ATSC or an Extended Event Descriptor in DVB. An empty string will be returned when no ETT is available for this event.

retrieveContentAdvisory (listener : org.atsc.abstractSI.util.SIRetrievalListener) : ContentRatingAdvisory[]

Returns a list of Content Advisory information for each Rating Region.

getRunningStatus () : short

Returns the running status of this event (see DVB SI documentation).

getTvChannel () : TvChannel

Returns the TvChannel this program event is associated with. In DVB, events and TvChannels are associated via the service ID; in ATSC, they are associated via the source ID.

getCost () : String

This method returns the cost of an IPPV Program Event or null if this is not an IPPV event.

getTheme () : short[]

This method returns a list of themes associated with the program. It is represented as a number and is

system-specific (e.g., DVB content nibble). Refer to the appropriate specification for details.

2.2.3 ContentRatingAdvisory 310

The Content Advisory is used to indicate, for a given event, ratings for any or all of the rating dimensions defined for each rating region. Ratings may be given for any or all of the defined regions. An Event without content advisory indicates that the rating value for any rating dimension defined in any rating region is zero. The absence of ratings for a specific dimension is completely equivalent to having a zero-valued rating for such a dimension. The absence of ratings for a specific region implies the absence of ratings for all the dimensions in the region.

Public Operations:

getRatingRegion () : short

Returns a unsigned 8-bit integer that specifies the rating region for which the data in this object is defined. The rating_region associates ratings data given here with data defined in a RRT tagged with the corresponding rating region.

Note that the DVB rating system is based on age only. It can be easily mapped to this more elaborate rating system as one of the dimensions.

getDimensions () : short[]

Returns a list of all dimensions rated for this Rating Region.

getRatingValue (ratedDimension : short) : short

Returns a number representing the rating value of the dimension specified by the parameter for this rating region.

getRatingText (ratedDimension : short) : String

Returns the rating description display string for the specified dimension. It shall be limited to 16 characters or less.

2.2.4 RatingRegion 330

This interface defines all rating dimensions of a specific rating region.

PSIP Rating defines the TV parental guideline system referenced by any content advisory descriptor for a Service or Event. It is based on the RRT.

Public Operations:

getNumberOfDimensions () : short

Returns the number of rating dimensions defined in this Rating Region.

getRegionName () : String

Returns the rating region name, e.g., "U.S. (50 states + possessions)", associated with the Rating Region. The display string for the rating region name shall be limited to 32 characters or less.

getRatingDimensions () : RatingDimension[]

Returns an array of all Rating Dimensions defined for this Rating Region.

2.2.5 RatingDimension 350

One dimension in the U.S. rating region, for example, is used to describe the MPAA list. The

dimension name for such a case may be defined as "MPAA".

Another example of a rating dimension may be a age-based DVB rating.

Public Operations:

getDimensionName () : String

Returns a string which represents the dimension name being described by this object, such as "MPAA". The dimension name display string shall be limited to 20 characters or less.

isGraduatedScale () : boolean

Indicates whether or not the rating values in this dimension represent a graduated scale, i.e., higher rating values represent increasing levels of rated content within the dimension. Value 1 means yes, while value 0 means no.

getNumberOfLevels () : short

Returns 4-bit field (1-15) specifying the number of values defined for this particular dimension.

getRatingLevelDescription (ratingLevelIndex : short) : String[]

Returns a pair of Strings describing the specified Rating Level for this Dimension.

The first string represents the abbreviated name for one particular rating value. The abbreviated name for rating value 0 shall be set to a null string, i.e., "". The abbreviated value display string shall be limited to 8 characters or less.

The second string represents the full name for one particular rating value. The full name for rating value 0 shall be set to a null string, i.e., "". The

rating value display string shall be limited to 150 characters or less.

2.2.6 TvChannelWithSchedule 300

This interface extends the TvChannelDetails by adding access to the program schedule associated with this TvChannel. It is derived from TvChannelDetails 245.

Public Operations:

getProgramSchedule () : ProgramSchedule

Returns a schedule of programs/events associated with this TvChannel.

2.3 Selection Package (FIG. 4)

This view at the SI is from the point of view of channel selection. It represents the information that is necessary to provide to other APIs, such as NetworkInterfaceController, JMF Player, A/V Decoder, etc., to select, tune and eventually decode a specific MPEG-2 service and its components.

2.3.1 ChannelComponent 410

This interface represents an abstraction of an MPEG Elementary Stream. It provides information about individual components of the TvChannel 225. It may be used by a Player to select the appropriate components of the TS.

Public Operations:

getComponentName () : String

Returns a name associated with this component. The Component Descriptor may be used if present. A

generic name (e.g., video, first audio, etc.) may be used otherwise.

getAssociatedLanguage () : int

Returns 3-byte (24 bits) field, based on ISO 639.2/B, specifies the language used for the elementary stream. In case of no language specified for this elementary stream, e.g., video, each byte shall have the value 0x00.

getStreamType () : short

Returns the stream type of this component.

See also StreamType

2.3.2 MPEGLocator 440

Derived from Locator 430.

Public Operations:

getNetworkID () : int

Called to determine the Network ID of the network this Locator represents.

getServiceID () : int

Called to determine the Service ID (MPEG program number) of the service this Locator represents.

getTransportStreamID () : int

Called to determine the TransportStreamID of the transport this locator represents.

getEventID () : int

Returns an identification of this program event.

Note: ATSC PSIP Event ID is unique only within a single EIT table while the DVB Event ID is unique within the service.

2.3.3 ATSCLocator 450

Derived from MPEGLocator 440.

Public Operations:

getSourceID () : int

Returns an integer number that identifies the programming source associated with the virtual channel. In this context, a source is one specific source of video, text, data, or audio programming.

Source ID value zero is reserved. Source ID values in the range 0x0001 to 0x0FFF shall be unique within the TS that carries the VCT, while values 0x1000 to 0xFFFF shall be unique at the regional level.

Values for source_ids 0x1000 and above shall be issued and administered by a Registration Authority designated by the ATSC.

2.3.4 MPEGChannelComponent 420

This is a specific Channel Component representing MPEG-2 elementary stream.

Derived from ChannelComponent 410.

Public Operations:

getPID () : short

Returns the PID the data of elementary stream is sent on in the TS.

getPcrPID () : short

Returns the PCR PID number associated with this component.

getTag () : int

Returns the component tag (Stream Identifier Descriptor) of this elementary stream, or null if none is present.

getAssociationTag () : int

Returns the association tag (Association Tag Descriptor) of this elementary stream, or null if none is present.

2.3.5 TunableChannelControl (415)

TunableChannelControl is an interface which accepts a TvChannel object that can be tuned to and played by the JMF Player or its DTV derivative.

Derived from JMF javax.media.Control.

Public Operations:

getCurrentChannel(): TvChannel

Returns the currently-playing TvChannel.

play(newChannel : TvChannel)

This method is called to initiate a tuning, decoding and presentation of a TvChannel.

play(newChannel : TvChannel, components : int[])

This method is called to initiate a tuning, decoding and presentation of a TvChannel. The additional parameter specifies which components of the TvChannel to play. Components are identified by their tag number.

play(newChannel : TvChannel, language : int)

This method is called to initiate a tuning, decoding and presentation of a TvChannel. The additional parameter specifies which components of the TvChannel to play based on the language association.

2.4 Descriptor Package (FIG. 5)

This package describes a general API mechanism of retrieving descriptors from any type of an MPEG table:

MPEG PSI, DVB SI, ATSC PSIP, or even private tables such as DCII.

The intention is to define an SI-format-independent mechanism of retrieving these descriptors to minimize the knowledge of DVB and ATSC differences and the need for special code in an application calling this API.

Refer to the discussion in section 1.4, "Descriptor View".

2.4.1 MPEGTableDescriptors 540

This interface provides a mechanism for retrieving MPEG Descriptors associated with any MPEG, DVB, ATSC or even a private table. It either returns a set of descriptors or a set of available descriptor tags. The calling application may also hint which descriptors it is interested in. Only a subset of those will be returned if they exist in the specified table.

Descriptors are primarily identified by a URL. This is in many cases not enough because a service, for instance, may be described in multiple tables, such as PMT and SDT. Therefore, table type is specified to identify which type of a table to retrieve descriptors from. In some rare cases an additional identification is needed. For instance, if a RRT type is specified, a Rating Region ID is needed to retrieve the proper subset of descriptors. The same applies to BAT type tables, where a Bouquet ID is needed to determine which descriptors to retrieve.

Additionally, an optional set of descriptor tags can be specified in the call to limit the search and the returned set of descriptors.

Derived from DescriptorTag 520 and TableType 510.

Public Operations:

retrieveDescriptors (url : Locator, tableType : short, someTags : short[], listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.descriptor.Descriptor[]

Retrieves a set of descriptors. This method retrieves all or a set of descriptors associated with the entity specified by the Locator (URL) delivered in the specified table in the order the descriptors are broadcast.

Parameter url - A URL-based specification of an entity (such as a TS, a service, etc.) for which to get the descriptors.

Parameter tableID - A TableType of the table from which to retrieve the specified descriptors.

Parameter someTags - A list of tags for descriptors (identified by their tags) the application is interested in. All non-applicable tag values are ignored. If this list is empty or null, all descriptors will be returned.

Parameter listener - A listener which receives the delivery event when data are retrieved asynchronously. If an application does not provide a listener (null), no asynchronous retrieval is attempted. The listener is registered for this one call only.

Return Value - A set (or subset) of Descriptor objects as indicated in someTags.

See also Descriptor.

retrieveDescriptors (url : Locator, tableType : short, entityID : short, someTags : short[], listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.descriptor.Descriptor[]

Retrieves a set of descriptors. This method retrieves all or a set of descriptors associated with the entity specified by the Locator (URL) delivered in the specified table in the order the descriptors are broadcast.

Parameter url - see previous definition.

Parameter tableID - see previous definition.

Parameter entityID - An ID representing a specific entity described in the specified table. The entity type depends on the table type. For example, if the table ID identifies an BAT, then the entity ID is a specific Bouquet ID.

Parameter someTags - see previous definition.

Parameter listener - see previous definition.

Return Value - see previous definition.

See also Descriptor.

retrieveDescriptorTags (url : Locator, tableType : short, listener : org.atsc.abstractSI.util.SIRetrievalListener) : short[]

Retrieves the tags of all descriptors associated with the entity specified by the Locator (URL) that are actually broadcast for the specified table type. The tags are returned in the same order as the descriptors are broadcast (i.e., in the transport stream).

Parameter url - see previous definition.

Parameter tableID - A Table ID of the table from which to retrieve the descriptor tags.

Parameter listener - see previous definition.

Return Value - The tags of the descriptors actually broadcast for the specified table (identified by their tags).

See also DescriptorTag.

retrieveDescriptorTags (url : Locator, tableType : short, entityID : short, listener : org.atsc.abstractSI.util.SIRetrievalListener) : short[]

Retrieves the tags of all descriptors associated with the entity specified by the Locator (URL) that are actually broadcast for the specified table type. The tags are returned in the same order as the descriptors are broadcast.

Parameter url - see previous definition.

Parameter tableID - see previous definition.

Parameter entityID - An ID representing a specific entity described in the specified table. The entity type depends on the table type. For example, if the table ID identifies a BAT, then the entity ID is a specific Bouquet ID.

Parameter listener - see previous definition.

Return Value - see previous definition.

See also DescriptorTag.

2.4.2 Descriptor 530

This interface specifies the basic structure of an MPEG Descriptor. It consists of a Tag, Length and a array of bytes.

Derived from SystemInformationType 810 and DescriptorTag 520.

Public Operations:

getTag () : short

Returns the descriptor tag.

getByteAt (index : int) : byte

Returns a particular byte within the descriptor content.

getLength () : short

Returns the length of the descriptor content.

getContent () : byte[]

Returns the whole descriptor content.

2.4.3 DescriptorTag 520

This interface defines constants corresponding to the most common descriptor tags. See also Descriptor.

Public Attributes:

NETWORK_NAME : short = 0x40

SERVICE_LIST : short = 0x41

STUFFING : short = 0x42

SATELLITE_DELIVERY_SYSTEM : short = 0x43

CABLE_DELIVERY_SYSTEM : short = 0x44

BOUQUET_NAME : short = 0x47

SERVICE : short = 0x48

COUNTRY_AVAILABILITY : short = 0x49

LINKAGE : short = 0x4A

NVOD_REFERENCE : short = 0x4B

TIME_SHIFTED_SERVICE : short = 0x4C

SHORT_EVENT : short = 0x4D

EXTENDED_EVENT : short = 0x4E

TIME_SHIFTED_EVENT : short = 0x4F

```
COMPONENT : short = 0x50
MOSAIC : short = 0x51
STREAM_IDENTIFIER : short = 0x52
CA_IDENTIFIER : short = 0x53
CONTENT : short = 0x54
PARENTAL_RATING : short = 0x55
TELETEXT : short = 0x56
TELEPHONE : short = 0x57
LOCAL_TIME_OFFSET : short = 0x58
SUBTITLING : short = 0x59
TERRESTRIAL_DELIVERY_SYSTEM : short = 0x5A
MULTILINGUAL_NETWORK_NAME : short = 0x5B
MULTILINGUAL_BOUQUET_NAME : short = 0x5C
MULTILINGUAL_SERVICE_NAME : short = 0x5D
MULTILINGUAL_COMPONENT : short = 0x5E
PRIVATE_DATA_SPECIFIER : short = 0x5F
SERVICE_MOVE : short = 0x60
SHORT_SMOOTHING_BUFFER : short = 0x61
FREQUENCY_LIST : short = 0x62
PARTIAL_TRANSPORT_STREAM : short = 0x63
DATA_BROADCAST : short = 0x64
AC3_AUDIO : short = 0x81
PROGRAM_IDENTIFIER : short = 0x85
CAPTION_SERVICE : short = 0x86
CONTENT_ADVISORY : short = 0x87
EXTENDED_CHANNEL_NAME : short = 0xA0
SERVICE_LOCATION : short = 0xA1
TIME_SHIFTED_ATSC_SERVICE : short = 0xA2
COMPONENT_NAME : short = 0xA3
```

2.4.4 TableType 510

This interface defines a set of constants corresponding to MPEG, ATSC and DVB table types, or private tables such as DCII.

Public Attributes:

UNKNOWN : short = 0

CA_INFO : short = 1

MPEG PSI CAT table.

SERVICE_INFO : short = 2

MPEG PSI PMT table.

TRANSPORT_INFO : short = 3

MPEG PSI TSDT table.

NAVIGATION_INFO : short = 4

DVB BAT and SDT tables, and ATSC VCT table.

NET_INFO : short = 5

DVB NIT table and ATSC MGT table.

RATING_INFO : short = 6

ATSC RRT table.

TIME_INFO : short = 7

DVB TOT table and ATSC STT table.

EVENT_INFO : short = 8

DVB EIT table and ATSC EIT table.

2.5 Pipeline Package (FIG. 6)

This package represents the pipeline (or network) view. It represents objects related to the transport delivery mechanism such as MPEG-2. It could also include information about non-MPEG delivery protocols such as IP.

2.5.1 SatelliteInformation 605

This interface represents information about Satellites in a satellite delivery network.

Public Operations:

getSatelliteID () : int

Returns the ID of this Satellite.

getSatelliteName () : String

Returns the full Satellite name.

getSatelliteAbbreviation () : String

Returns the abbreviated satellite name.

getNumberOfTransponders () : int

Returns the number of Transponders available on this Satellite.

getTransponderNumbers () : int[]

This method returns an array of transponder numbers available on this Satellite.

getTransponderName (tranponderNumber : int) : String

Returns the name of the specified Transponder.

getTranponderNumber (transportStreamID : int) : int

Returns the transponder number that delivered the specified TS.

2.5.2 BouquetInformation 630

This interface represents information about a bouquet (a collection of services which can span TS and network boundaries), which is a DVB-specific concept.

Public Operations:

getBouquetID () : int

This method returns the ID of this Bouquet definition.

getName () : java.lang.String

This method returns the name of this Bouquet.

2.5.3 TransportStreamInfo 660

This interface provides information about a TS.

Derived from SystemInformationType 810.

Public Operations:

getLocator () : org.davic.net.Locator

This method returns the URL of this TS.

getNetworkID () : int

Returns the ID of the Network which carries this TS.

getTransportStreamID () : int

This method returns the ID of this TS.

getDescription () : String

Returns a textual name or a description of this TS.

getOriginalNetworkID () : int

This method returns the Network ID of the Network where this TS originated from.

This method returns the same ID as getNetworkID if this TS originated on the Network that carries it.

2.5.4 NetworkInformation

This interface provides descriptive information about a Network of Transport Streams.

Public Operations:

getNetworkID () : int

This method returns the ID of this Network

getLocator () : org.davic.net.Locator

This method returns the URL of this Network.

getNetworkName () : java.lang.String

This method returns the name of this network.

2.5.5 MPEGPipeline 670

The MPEG SI Manager represents MPEG-2 specific SI. Derived from PipelineInformation 650 and MPEGTableDescriptors 540.

Public Operations:

retrieveSatelliteInformation (tranportStreamID : int, listener : org.atsc.abstractSI.util.SIRetrievalListener) : SatelliteInformation[]

This method returns an array of object representing information about the Satellites carrying the specified TransportStream.

If no TS is specified, it returns an array of objects representing information about all known Satellites.

It returns an empty array if this is not a satellite network.

retrieveBouquet (bouquetID : int, listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.pipeline.BouquetInformation

This method returns information about the specified Bouquet.

retrieveBouquets (listener : org.atsc.abstractSI.util.SIRetrievalListener) : BouquetInformation[]

Returns an array of BouquetInformation objects representing all known Bouquets.

retrieveTransportStream (url : Locator, listener : org.atsc.abstractSI.util.SIRetrievalListener) : TransportStreamInfo

The method returns a specific TransportStreamInfo object representing information about the specified TS.

retrieveTransportStreams (networkID : int, listener : org.atsc.abstractSI.util.SIRetrievalListener) : TransportStreamInfo[]

Returns an array of TransportStreamInfo objects representing all known Transport streams for the specified network.

retrieveNetwork (networkID : int, listener : org.atsc.abstractSI.util.SIRetrievalListener) : NetworkInformation

Returns the NetworkInformation specified by its ID.

retrieveNetworks (listener : org.atsc.abstractSI.util.SIRetrievalListener) : NetworkInformation[]

Returns an array of NetworkInformation objects representing all known networks.

addListener (listener : SIChangeListener) : void

Called to register a SIChangeListener 815 for changes related to changes in objects provided by the SIManager 240.

This includes TransportStreamInfo 660, NetworkInformation 610, etc. but excludes TvChannel 225 and ProgramEvent 340 which can be listened to via the appropriate collections, such as the ChannelCollection 220 and ProgramSchedule 320. Therefore, the

SIInfoChangeEvent 850 will be delivered to the listener.

removeListener (listener : SIChangeListener) : void

Called to deregister a SIChangeListener 815.

2.5.6 SISpecificManager 640

This SI manager has specific information about the content delivery media. Derived from SIManager 240.

Public Operations:

getPipelineInfo () : org.atsc.abstractSI.pipeline.PipelineInformation[]

This method returns a list of objects representing different content delivery media, such as MPEG-2 transport.

2.5.7 PipelineInformation 650

This abstract class is a placeholder for different content delivery media. It must be extended by an object that is specific to a particular transport mechanism, such as MPEG-2 TS.

2.6 Data Package (FIG. 7)

This package represents a schedule of data events. Similar to audio/video programs, there may also be data events scheduled for a given TvChannel.

2.6.1 TvChannelWithData 710

This interface extends the TvChannelDetails by adding access to the data event schedule associated

with this TvChannel. Derived from TvChannelDetails 245.

Public Operations:

getDataSchedule () : org.atsc.abstractSI.data.DataSchedule

This method returns a schedule of data events.

2.6.2 DataSchedule 720

This interface represents a collection of data events for a given TvChannel ordered by time.

Public Operations:

addListener (listener : org.atsc.abstractSI.util.SIChangeListener) : void

Called to register an SIChangeListener 815 for events related to changes of the DataEvents 730 on this schedule. The DataChangeEvent 840 is delivered to the specified listener when any DataEvent 730 on this schedule changes.

removeListener (listener : org.atsc.abstractSI.util.SIChangeListener) : void

Called to deregister a SIChangeListener.

retrieveEvent (when : java.util.Date, listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.data.DataEvent

Returns the data event for the specified time. The data event which contains the specified time will be returned. The specified time falls between the data event's start time and the start time plus the event's duration.

Null is returned when the specified time does not fall inside any known data event.

retrieveEvents (startOfInterval : java.util.Date, endOfInterval : java.util.Date, listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.data.DataEvent[]

Returns all known data events on this channel for the specified time interval.

retrieveEvent (locator : org.davic.net.Locator, listener : org.atsc.abstractSI.util.SIRetrievalListener) : DataEvent

This method retrieves a data event matching the locator. The event must be part of this schedule.

2.6.3 DataEvent 730

This object represents a data event associated with a TvChannel 225 for a particular time interval. In ATSC, the information is delivered in DIT tables. DIT is an extension of PSIP. It is similar to the EITs, but it announces data events (not audio-visual events) and is defined in the ATSC T3/S13 Data Broadcast Specification.

Derived from SIUpdate 830.

Public Operations:

getLocator () : org.davic.net.Locator

Returns a Locator representing this data event.

getStartTime () : java.util.Date

Retuns the start time of this data event.

getEndTime () : java.util.Date

Retuns the end time of this data event.

getDuration () : long

Returns the duration of this data event in seconds.

getTitle () : java.lang.String

Returns the data event title.

retrieveDescription (listener : org.atsc.abstractSI.util.SIRetrievalListener) : java.lang.String

Returns a textual description of the event. This information comes from the ETT in ATSC, or an Extended Event Descriptor in DVB. An empty string will be returned when no ETT is available for this event.

retrieveContentAdvisory (listener : org.atsc.abstractSI.util.SIRetrievalListener) : org.atsc.abstractSI.guide.ContentRatingAdvisory

Returns a list of Content Advisory information for each Rating Region.

getTvChannel () : org.atsc.abstractSI.navigation.TvChannel

Returns the TvChannel this data event is associated with. In DVB, events and TvChannels are associated via the service ID; in ATSC, they are associated via the source ID.

2.7 Utility Package (FIG. 8)

This package defines classes and interfaces that provide support functions to the SI packages. This includes the notification mechanism (events and listeners), the Factory Method for creating the SIManager object, and all Exceptions.

The Factory Method is a methodology and structure for solving a problem, as known from the field of object-oriented programming.

2.7.1 SIFactory 805

This class provides a mechanism to create objects that implement the SIManager interface 240. This class is modeled after the Factory Method design pattern.

Public Operations:

SIFactory () :

Constructor

getSIManager () : SIManager

Returns an implementation of the SIManager interface or null if not available.

2.7.2 SIChangeListener 815

The SIChangeListener interface shall be implemented by using application classes to listen to changes in SI objects. It provides a method to be called back by the listened to SI object to notify of an event.

Public Operations:

SIChange (event : SIChangeEvent) : void

This method gets called when an existing SI object is changed, a new SI object is detected or an existing SI object is no longer available.

2.7.3 SIChangeEvent 834

SIChangeEvent objects 834 are sent to SIChangeListeners 815 to notify of a new event.

Derived from EventObject 820.

Public Operations:

getType () : byte

Returns the event type (the possible values are defined in the SIChangeEventType interface).

2.7.4 SIChangeEventType

This interface defines the constants corresponding to the SIChangeEvent type values.

Public Attributes:

OBJECT_CHANGED : byte = 1

An existing object has changed.

NEW_OBJECT : byte = 2

New object detected.

OBJECT_UNKNOWN : byte = 3

Object no longer available.

2.7.5 TvChannelChangeEvent 835

This event delivers information about a particular TvChannel. Derived from SIChangeEvent 834.

Public Operations:

getChangedChannel () : Locator

This method is called to determine which channel has changed.

2.7.6 ProgramChangeEvent 845

This event delivers information about a particular ProgramEvent.

Derived from SIChangeEvent 834.

Public Operations:

getChangedProgram () : Locator

This method is called to determine which ProgramEvent has changed.

2.7.7 SIInfoChangeEvent 850

This event delivers information about a particular high-level SI object usually obtained via the SI Manager. Derived from SIChangeEvent 834.

Public Operations:

getSIObjectType () : short

This method is called to determine which type of an SI object has changed. This may be TransportStreamInfo 660, NetworkInformation 610, BouquetInformation 630, RatingRegion 330, etc.

getSIObjectID () : int

This method returns the ID of the changed SI object whose type is identified by the getSIObjectType method. For example, if the SIObjectType is TransportStreamInfo, then the SIObjectID will be the TS ID.

2.7.8 SIUpdate 830

This interface can be associated with any SI entity. It provides information about when the data was last updated. The SI database may have new information which all listeners get notified about.

Public Operations:

getUpdateTime () : java.util.Date

Return the time when the information contained in the object that implements this interface was last updated.

Return Value - The date of the last update.

2.7.9 SIDelayedDeliveryException 875

This exception is thrown when requested data is not available immediately (e.g., not cached) and signals an asynchronous delivery of the data.

Derived from Exception 860.

Public Operations:

getRequest () : org.atsc.abstractSI.util.SIRequest

This method returns an object representing the asynchronous request. It can be used to cancel the request and to associate it with the event delivering the requested data.

2.7.10 SIRetrievalListener 870

This interface shall be implemented by application classes to receive events about completion of asynchronous SI requests.

In general, the listener registers itself at the time of the potentially asynchronous call (as one of the parameters). The listener is registered for one call only and is automatically deregistered when the request is satisfied.

Public Operations:

postRetrievalEvent (event : org.atsc.abstractSI.util.SIRetrievalEvent) : void

This method is called to deliver an asynchronous SI retrieval event to the listener.

2.7.11 SIRetrievalEvent 865

This event delivers data requested asynchronously by any of the "retrieveXXX" calls. All the methods

which start with the word "retrieve" fall into this category, e.g., retrieveDetails() in block 225, retrievePresentEvent() in block 320, etc.

It either delivers the data itself or an indication of a failure. It also provides mechanism to associate this event with the original request using the request sequence number.

This event is delivered only when the "retrieveXXX" call throws the SIDelayedDelivery exception 875, which contains the SIRequest object 886 with the request sequence number.

Derived from EventObject 820.

Public Operations:

getSequenceNumber () : int

This method returns the sequence number assigned to the original asynchronous retrieval request to which this event is responding.

2.7.12 SIRequest 886

This object is used to facilitate asynchronous retrieval of SI data. This object can be used to cancel a pending request and to associate the request with a event delivering the requested data.

Public Operations:

cancelRequest () : boolean

This method will cancel a pending request.

Return Value 'True' indicates a successful cancellation of this request. 'False' indicates that the request has already been delivered and cannot be cancelled.

getSequenceNumber () : int

This method returns a number associated with this asynchronous retrieval call. It can be used to pair the subsequent event with this request.

2.7.13 SIRetrievalSuccessEvent 884

This event signals that the requested data has been retrieved and delivered.

Derived from SIRetrievalEvent 865.

2.7.14 SIRetrievalSingleSuccessEvent 892

The event delivers a single object that was requested. Derived from SIRetrievalSuccessEvent 884.

Public Operations:

getResult () : java.lang.Object

This method is used to obtain the data delivered by this event.

The specific type (class) of the returned object can be anticipated from the context of the original call: it is the same object that would be retutned by the "retrieveXXX" method synchronously.

2.7.15 SIRetrievalMultipleSuccessEvent 894

The event delivers a array of objects that were requested. Derived from SIRetrievalSuccessEvent 884.

Public Operations:

getResult () : java.lang.Object[]

This method is used to obtain the data delivered by this event.

The specific type (class) of the returned objects can be anticipated from the context of the original

call: it is the same object that would be returned by the "retrieveXXX" method synchronously.

2.7.16 SIRetrievalFailEvent 882

This event signals a failure to deliver the requested data. Possible reasons for a failure might be that the data is not present in the TS or resources were not available to obtain the data.

Derived from SIRetrievalEvent 865.

2.7.17 SIException 880

This is a generic exception which can be thrown when a particular SI-related call contains invalid parameters. Derived from Exception 860.

2.7.18 SystemInformationType 810

This interface provides access to specific SI databases (e.g., DVB SI database, ATSC PSIP database, etc.)

Public Attributes:

ATSC_PSIP : short = 1

DVB_SI : short = 2

SCTE_SI : short = 3

Public Operations:

getSystemInformationType () : short

Called to determine the specific SI format this element was delivered in (e.g., ATSC PSIP, DVB SI, etc.).

2.7.19 DataChangeEvent 840

This event delivers information about a particular data event. Derived from SIChangeEvent 834.

Public Operations:

getChangedData () : org.davic.net.Locator

2.7.20 SINotAvailableException 890

The exception indicates that the requested data is not available for the particular instance.

Derived from SIException 880.

2.7.21 DeliverySystemType 807

This interface provides information about the delivery system type (e.g., cable, satellite, etc.) of the particular object implementing this interface.

Public Attributes:

CABLE_DELIVERY_SYSTEM : short = 1

SATELLITE_DELIVERY_SYSTEM : short = 2

TERRESTRIAL_DELIVERY_SYSTEM : short = 3

Accordingly, it can be seen that the present invention provides an API that allows applications at a digital television terminal to recover SI from a digital TS without regard to the specific format type. The API abstracts the relevant portions of the SI to provide it in a format that is usable by the terminal.

The system is suitable for use, e.g., with SI formats including MPEG PSI, DVB SI, and ATSC PSIP, and private SI.

Although the invention has been described in connection with various specific embodiments, those skilled in the art will appreciate that numerous

adaptations and modifications may be made thereto without departing from the spirit and scope of the invention as set forth in the claims.

For example, while various syntax elements have been discussed herein, note that they are examples only, and any syntax may be used.

Moreover, the invention is suitable for use with virtually any type of network, including cable or satellite television broadband communication networks, local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), internets, intranets, and the Internet, or combinations thereof.

Additionally, known computer hardware, firmware and/or software techniques may be used to implement the invention.